The International Workshop on Testing Database Systems

dbtest.io

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Adding Velocity to BigBench

[Work-in-Progress]

Todor Ivanov
(todor@dbis.cs.uni-frankfurt.de),
Patrick Bedué, Roberto V. Zicari
Frankfurt Big Data Lab,
Goethe University Frankfurt,
Germany

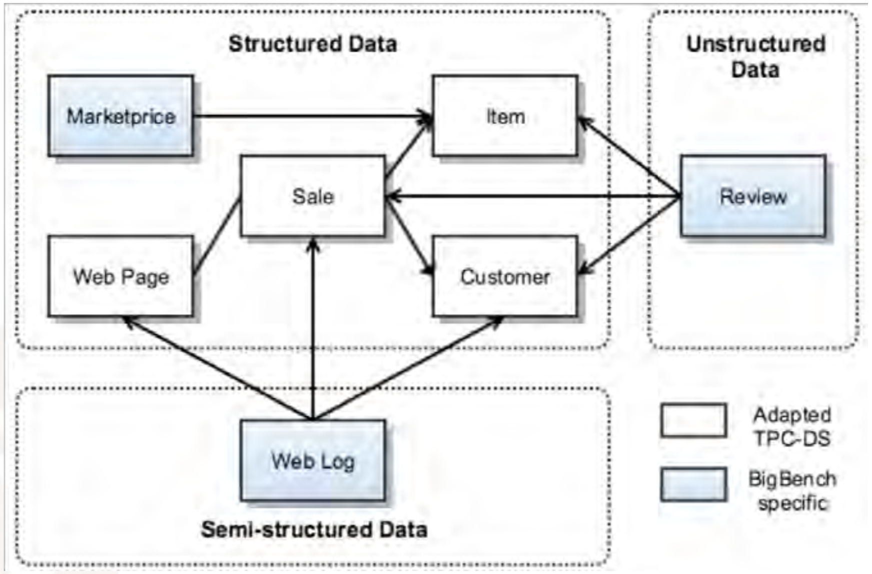
Ahmad Ghazal
Futurewei Technologies Inc.
Santa Clara, CA, USA

Content

1. Background BigBench
2. Motivation
3. Streaming Extension
4. Proof of Concept
5. Conclusions & Next Steps

BigBench [Ghazal et al. 2013] (presented @SIGMOD 2013)

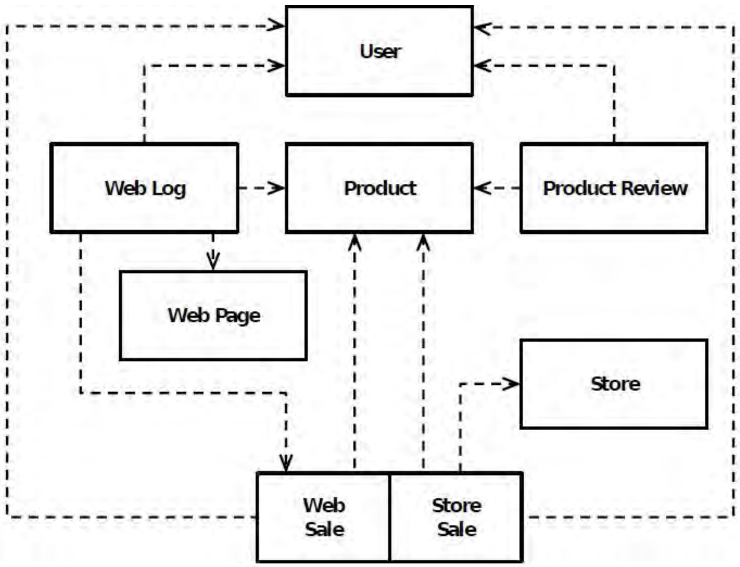
- End-to-end, technology agnostic, application-level Big Data benchmark.
  - On top of TPC-DS (decision support on retail business)



  - Adding semi-structured and unstructured data.
  - ***Focus on***: Parallel DBMS and MR engines (Hadoop, etc.).
  - ***Workload***: 30 queries
    - Based on big data retail analytics research
    - 11 queries from TPC-DS
- Adopted by TPC as TPCx-BB (http://www.tpc.org/tpcx-bb/). Implementation in HiveQL and Spark MLlib.

BigBench V2 [Ghazal et al. 2017] (presented @ ICDE 2017)

- BigBench V2 - a major rework of BigBench
  - Separate from TPC-DS and takes care of ***late binding***.
- New simplified data model and late binding requirements.
  - Custom made scale factor-based data generator for all components.



- 1 – many relationship : ------>
- **Semi-structured : key-value WebLog**
- Un-structured: Product Reviews

- Workload:
  - All 11 TPC-DS queries are ***replaced*** with new queries in BigBench V2.
  - New queries with similar business questions - ***focus on analytics on the semi-structured web-logs***.

Motivation

- Growing number of ***industry scenarios*** requiring streaming and ***new streaming engines:***

- New functionalities combining analytical with streaming features
  - Spark Structured Streaming
  - Calcite adapted by Flink SQL, Samza SQL, Drill, etc.
  - Kafka Streaming SQL - KSQL

- Need of standardized end-to-end application benchmarks covering all Big Data characteristics including velocity:
  - ***micro-benchmarks***: StreamBench, HiBench, SparkBench
  - ***application benchmarks:*** Linear Road, AIM Benchmark, Yahoo Streaming Benchmark, RIoTBench

→ none of the above benchmarks integrates an ***end-to-end real-world scenario*** implementing a Big Data architecture ***integrating storage, batch and stream processing components***

Our Requirements

- Create ***configurable*** data stream to simulate multiple scenarios:
  - real-time monitoring and dashboards (refresh rate in ***less than 3 seconds***)
  - streaming hours of ***history data for batch processing***

- Create ***deterministic*** data stream to:
  - ***compare accurately*** systems under test
  - ***validate and verify*** the workload results

- ***Isolate the stream engine execution*** as much as possible to avoid any external influence/bottlenecks, for example by the stream generation.

- ***Preserve*** the current BigBench specification, architecture, workload execution and metric.

Streaming Methodology (I)

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- ***Web-logs*** are key-value pairs representing user clicks (***JSON file***), for example:

{"wl_id":845, "wl_webpage_name":"webpage#20", "wl_item_id":758, "wl_timestamp":"2013-01-01 01:17:37", "wl_key$_1$":"value$_1$", "wl_key$_2$":"value$_2$", ..., "wl_key$_{100}$":"value$_{100}$"}

- ***Web-sales*** example:

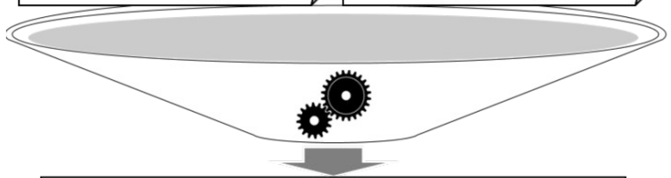
20|0|411|2|17.82|2013-01-27 16:12:32

- Web-logs and web-sales are ***generated in session window manner***.

- ***Sort*** the entries according to the ***event timestamp and create data windows*** depending on the simulated scenario.

weblogs
show webpage#01
show webpage#03
show webpage#04
show webpage#01
show webpage#02

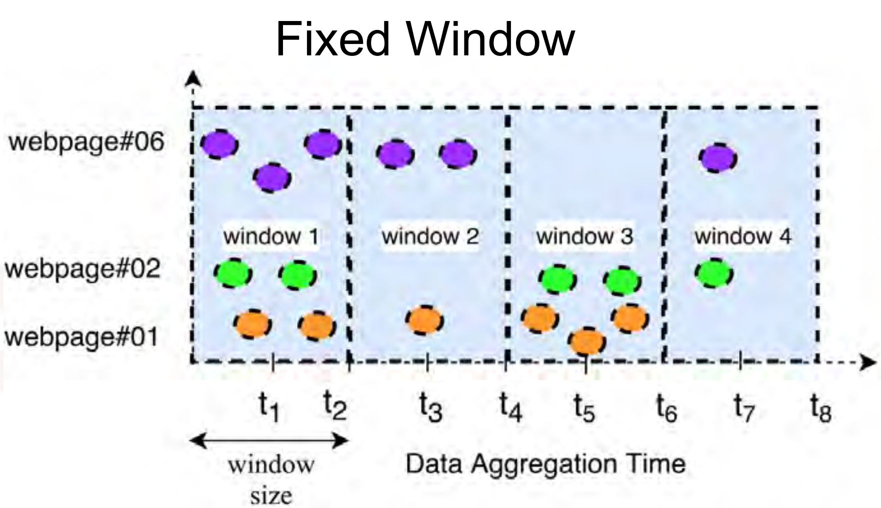
websales
buy item 34
buy item 7
buy item 151
...

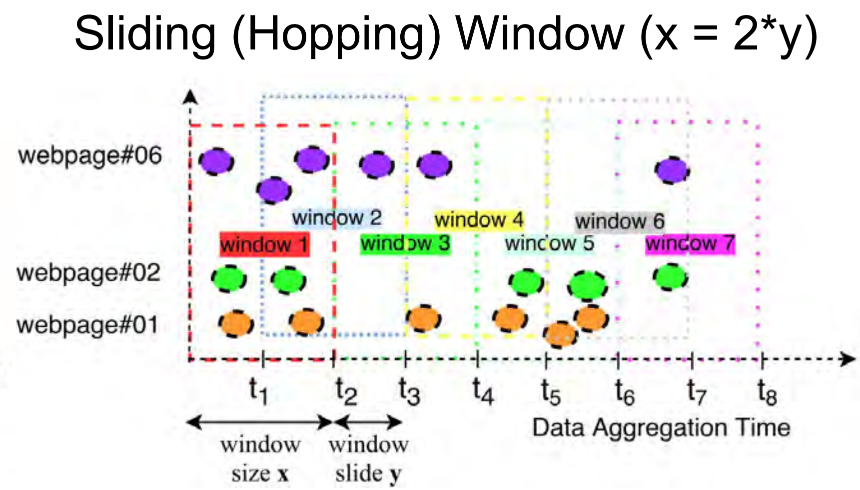


time	weblogs	websales
04:12:13	show webpage#01	
04:12:19	show webpage#03	
04:12:21	show webpage#04	
04:12:21		buy item 34
04:55:47	show webpage#01	
04:56:13	show webpage#02	
04:56:13		buy item 7
...	...	...

Streaming Methodology (II)

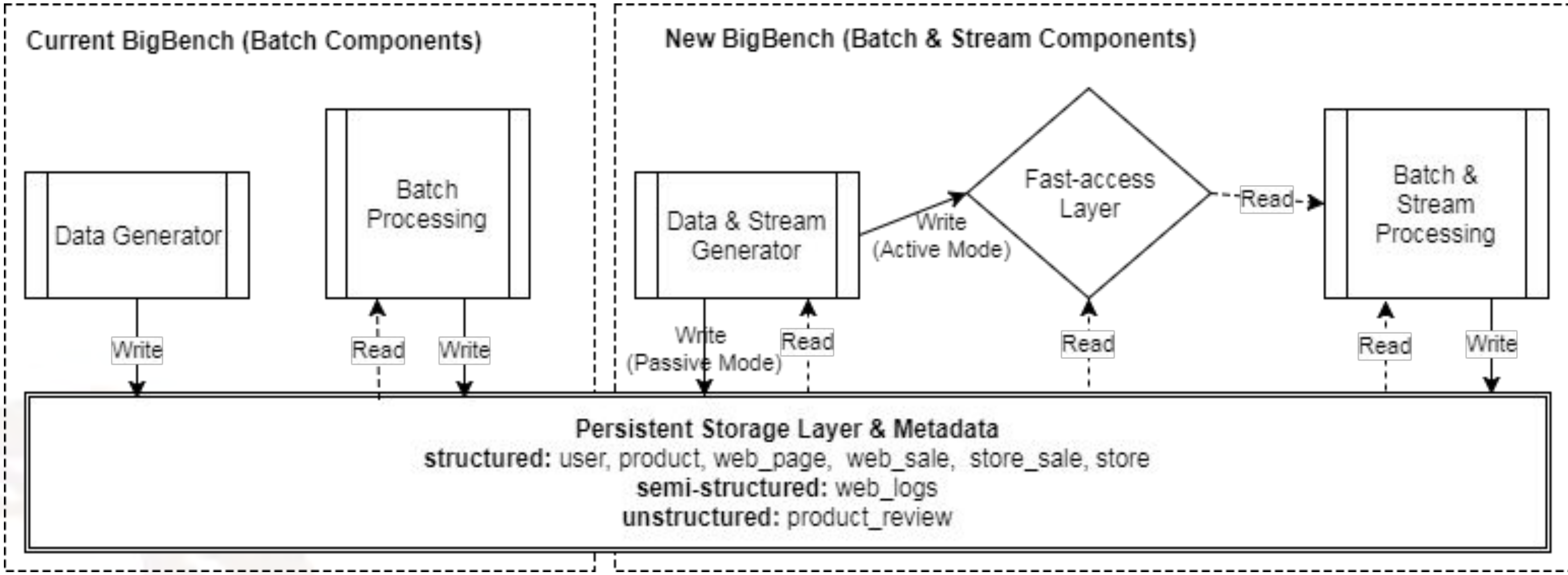
- Support for two window types:





- Configurable window parameters:
  - window size (x)
  - window slide (y) (e.g., hourly windows, starting every 30 minutes)
  - total runtime

Design Overview

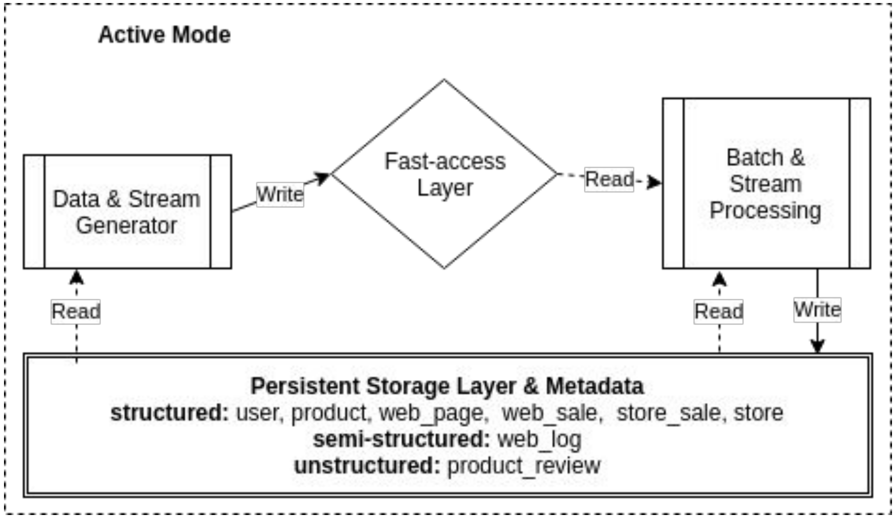


- Adding 3 new components:
  - ***Stream Generator***
  - ***Fast-access Layer***
  - ***Stream Processing***

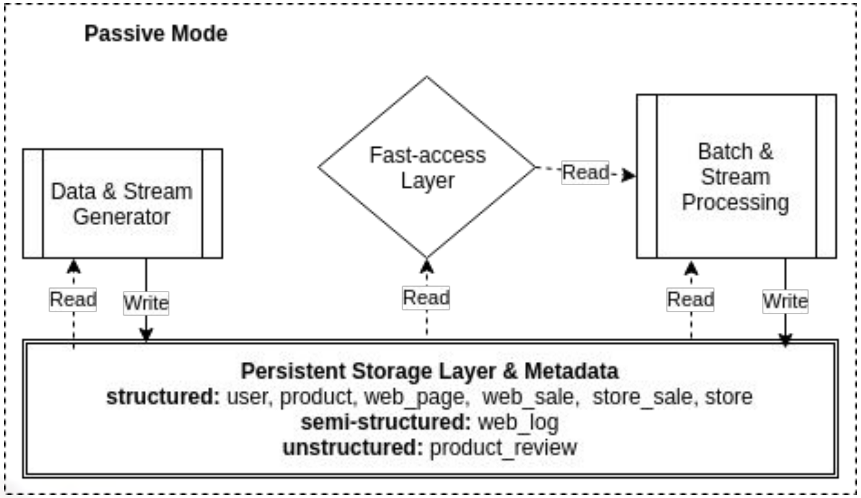
- Support for 2 stream execution modes:
  - ***Active Mode*** - simulate ***real-time*** data streaming (in second ranges)
  - ***Passive Mode*** - simulate ***data ingestion and transformation on*** micro-batch processing (in hour ranges)

Active and Passive Streaming Modes

- Active mode: ***parallel execution of the data stream generation*** and ***the actual stream processing.***



- Passive mode: ***sequential execution of data stream generation*** and ***the actual stream processing***.



Workloads

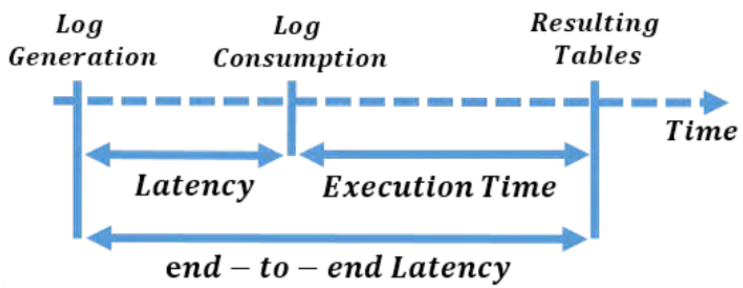
- The streaming workload consists of ***five queries*** executed periodically on a stream of data (web-logs and web-sales), covering simple aggregation and pattern detection operations:

  - $\mathbf{Q_{S1}}$: Find the 10 most browsed products in the last 120 seconds.
  - $\mathbf{Q_{S2}}$: Find the 5 most browsed products that are not purchased across all users (or specific user) in the last 120 seconds.
  - $\mathbf{Q_{S3}}$: Find the top ten pages visited by all users (or specific user) in the last 120 seconds.
  - $\mathbf{Q_{S4}}$: Show the number of unique visitors in the last 120 seconds.
  - $\mathbf{Q_{S5}}$: Show the sold products (of a certain type or category) in the last 120 seconds.

Metrics & Result Validation

- ***Execution time*** is the time between ***start and end*** of the query execution against the streaming data.

- ***End-to-end streaming execution time*** (Latency) - starting from the Stream Generator and stopping at the point where the data result is produced.



- Result ***validation based on scale factor*** similar to current BigBench validation (SF1):
  1. ***Store persistently*** the results of every query execution over a streaming window.
  2. ***Compare*** the results against the golden result once the benchmark run is finished.

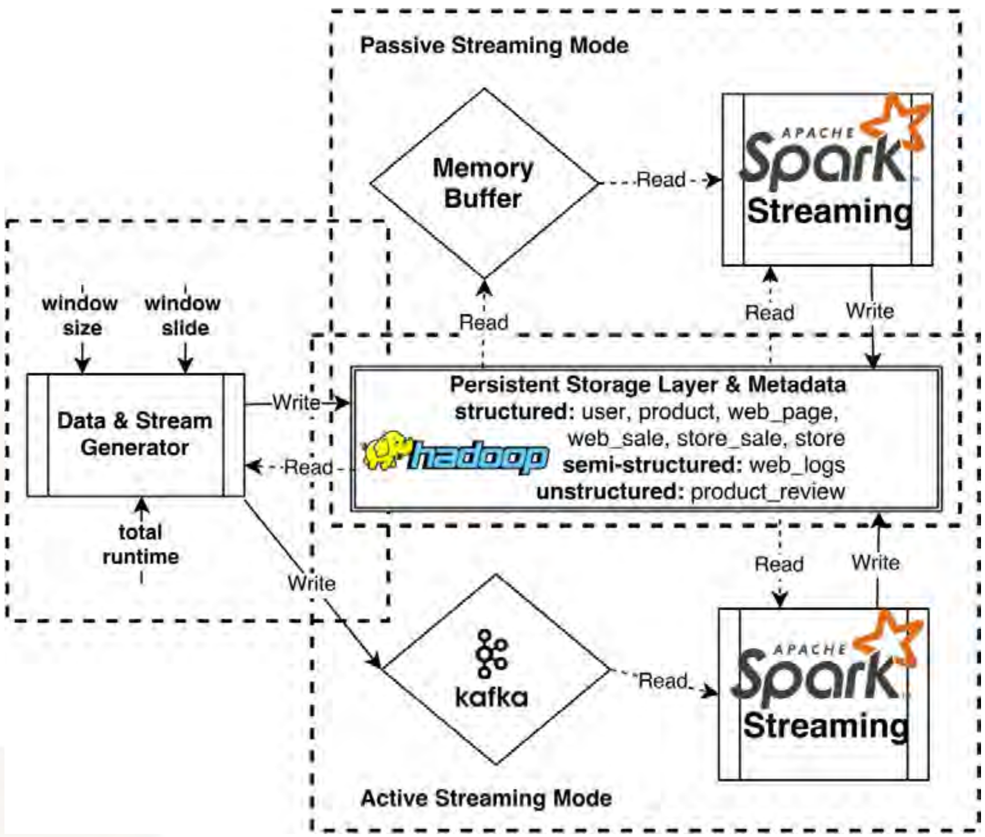
Proof of Concept Implementation

Active Mode Components:

- Stream Generator in **Spark**
- Persistent Storage Layer in **HDFS**
- <u>Fast-access Layer in **Kafka**</u>
- Stream Processing in **Spark Streaming**

Passive Mode Components:

- Stream Generator in **Spark**
- Persistent Storage Layer in **HDFS**
- <u>Fast-access Layer as **In-memory Buffer**</u>
- Stream Processing in **Spark Streaming**



Conclusion

- We present a **stream processing extension** of the BigBench benchmark.

- Our approach proposes **configurable active and passive streaming modes** in order to cover the different streaming requirements (ranging from seconds to hours).

- It supports **fixed and sliding window streaming** to better address the common data streaming use cases.

Features	**Active Mode**	**Passive Mode**
Fast-access Layer	Kafka	In-memory
Throughput	Low	High
Processing Type	Real-time	Batch
Performance Metrics	Inaccurate	Accurate
Scalability	Complex	Simple
Streaming and Processing	Parallel	Sequential

Next Steps

- New implementation on Spark Structured Streaming replacing Spark Streaming.

- Adding other engines such as Flink and Samza.

- Extending the coverage of the stream SQL operators (new workloads) including clustering, pattern detection and machine learning.

- Support for:
  - sliding windows in active mode
  - out-of-order record processing within and outside of a window
  - parallel query execution

- Validation experiments on a large-scale cluster with different active and passive mode architectures.

Thank you for your attention!

Acknowledgments. This work has been partially funded by the European Commission H2020 project DataBench - Evidence Based Big Data Benchmarking to Improve Business Performance, under project No. 780966. This work expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this work. The authors thank all the participants in the project for discussions and common work.

www.databench.eu

References

[Ghazal et al. 2013] Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. 2013. BigBench: Towards An Industry Standard Benchmark for Big Data Analytics. In SIGMOD 2013. 1197–1208.

[Ghazal et al. 2017] Ahmad Ghazal, Todor Ivanov, Pekka Kostamaa, Alain Crolotte, Ryan Voong, Mohammed Al-Kateb, Waleed Ghazal, and Roberto V. Zicari. 2017. BigBench V2: The New and Improved BigBench. In ICDE 2017, San Diego, CA, USA, April 19-22.

Backup Slides

Q_{S1} (HiveQL Q5 in BigBench V2)

Find the 10 most browsed products in the last 120 seconds.

```
SELECT wl_item_id, COUNT(wl_item_id) as cnt
FROM web_logs
WHERE wl_item_id IS NOT NULL
GROUP BY wl_item_id
ORDER BY cnt DESC LIMIT 10;
```

Q_{S2} (HiveQL Q6 in BigBench V2)

Find the 5 most browsed products that are not purchased across all users (or specific user) in the last 120 seconds.

```
SELECT wl_item_id AS br_id, COUNT(wl_item_id) AS br_count
FROM web_logs
WHERE wl_item_id IS NOT NULL
GROUP BY wl_item_id;
view_browsed.createOrReplaceTempView("browsed");

SELECT ws_product_id AS pu_id
FROM web_logs
WHERE ws_product_id IS NOT NULL
GROUP BY ws_product_id;
view_purchased.createOrReplaceTempView("purchased");

SELECT br_id, COUNT(br_id)
FROM browsed LEFT JOIN purchased ON browsed.br_id = purchased.pu_id
WHERE purchased.pu_id IS NULL
GROUP BY browsed.br_id LIMIT 5;
```

Q_{S3} (HiveQL Q16 in BigBench V2)

Find the top ten pages visited by all users (or specific user) in the last 120 seconds.

```
SELECT wl_webpage_name, COUNT(wl_webpage_name) AS cnt
FROM web_logs
WHERE wl_webpage_name IS NOT NULL
GROUP BY wl_webpage_name
ORDER BY cnt DESC LIMIT 10;
```

Q_{S4} (HiveQL Q22 in BigBench V2)

Show the number of unique visitors in the last 120 seconds.

```
SELECT COUNT(DISTINCT wl_customer_id) AS uniqueVisitors
FROM web_logs
WHERE wl_customer_id IS NOT NULL
ORDER BY uniqueVisitors DESC LIMIT 10;
```

Q_{S5} HiveQL

Show the sold products (of a certain type or category) in the last 120 seconds.

```
SELECT ws_product_id, COUNT(ws_product_id)
FROM web_sales
WHERE ws_product_id IS NOT NULL
GROUP BY ws_product_id
ORDER BY COUNT(ws_product_id) DESC LIMIT 10;
```